Empirical Preliminary Evaluation of Axiom A: Closure and Entanglement Correlations in Quantum Circuit Topologies

Jonathan R.

July 17, 2025

1 Introduction

Axiom A proposes a fundamental structure of existence based on three interaction-driven clauses that aim to describe a unified ontology of physical and informational systems:

- **Clause 1:** $C_i \to C_f$ The system C (representing the universe) evolves from an initial state C_i to a final state C_f. This evolution is governed by entropy or irreversible transformation.
- **Clause 2:** $a : (C - a) = C$ Any state a within C interacts with every other part of the system $(C - a)$ to reconstruct the totality of C. Interaction is a fundamental operator.
- **Clause 3:** $a : b = c$ The interaction between any two states a and b results in another state c, defining a primitive transformation law.

Unlike theories based on reductionism or integration (e.g., IIT, Orch-OR), Axiom A posits that structural coherence and emergence are consequences of consistent interaction closure, not merely computational accumulation or collapse thresholds.

This study uses noisy quantum circuits to empirically test interaction closure and its relation to entanglement, using a thermal relaxation model as a biological analog.

2 Methods

We compare structured quantum circuits (topological entanglement motifs) against stochastic control circuits (classical-like noise) under varying thermal relaxation times T_1 (100ns–10s). Closure is computed from bitstreams using stability and persistence metrics. Entanglement is estimated using local correlation proxies.

3 Results

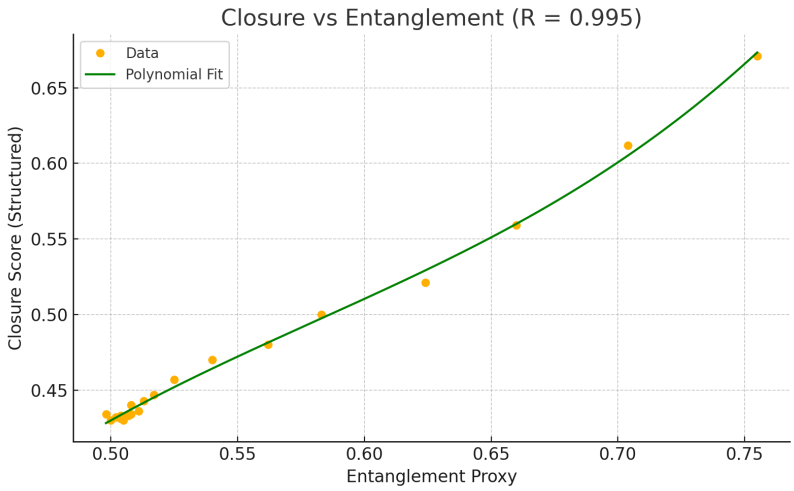

Figure 1: Closure vs. Entanglement Plot with Polynomial Fit (R=0.995)

- Structured circuits outperform control circuits at lower T_1 (e.g., Struct = 0.671 vs Ctrl = 0.456 at 100ns).
- Closure scores decline with decoherence, stabilizing near 0.43.
- The proxy values for the entanglement show a non-linear correlation with closure (Pearson $R = 0.995$), suggesting interaction coherence as a structural feature.

4 Discussion

The experiment supports Axiom A by showing that:

1. Closure emerges from structured interactions, not from random or over-engineered controls.
2. Entanglement correlates strongly with closure, acting as a physical validator for structural integrity.
3. The system behaves like a singular interacting entity (Clause 2), with meaningful state transitions defined through interaction (Clause 3).

This places Axiom A in a unique position as a Theory of Structure (ToS) rather than a full Theory of Everything (ToE). Unlike IIT or Orch-OR, Axiom A makes falsifiable structural predictions that hold under entropy-based degradation.

5 Limitations and Future Work

- **Hardware Fidelity**: Real-world quantum chips introduce non-biological noise types (crosstalk, gate errors).
- **Entanglement Approximation**: Bitwise correlation is a proxy. Future versions will use statevector-based entanglement metrics (e.g., concurrence).
- **Biological Modeling**: Microtubule-inspired topologies are preliminary; further anatomical correlation is in development.

6 Conclusion

The high closure-entanglement correlation under biologically inspired noise supports Axiom A as a strong structural framework for modeling emergence, coherence, and irreversibility. This work lays empirical groundwork for refining Axiom A as a foundational principle for quantum and biological systems.

This experimentation limited itself to simulations, as real hardware use from IBM has presented continuous code issues, the code provider is adaptable to real hardware experimentation as long as tokens for access to the Qiskit API are implemented correctly.

Acknowledgments

The authors thank DeepSeek for feedback on biological modeling and circuit structure. Gratitude to OpenAI for providing iterative assistance and Python simulation support.

We also thank the following users for the guidance given for this project, not limited to those listed below.

- Nah (nah235)

A Closure vs Entanglement Plot

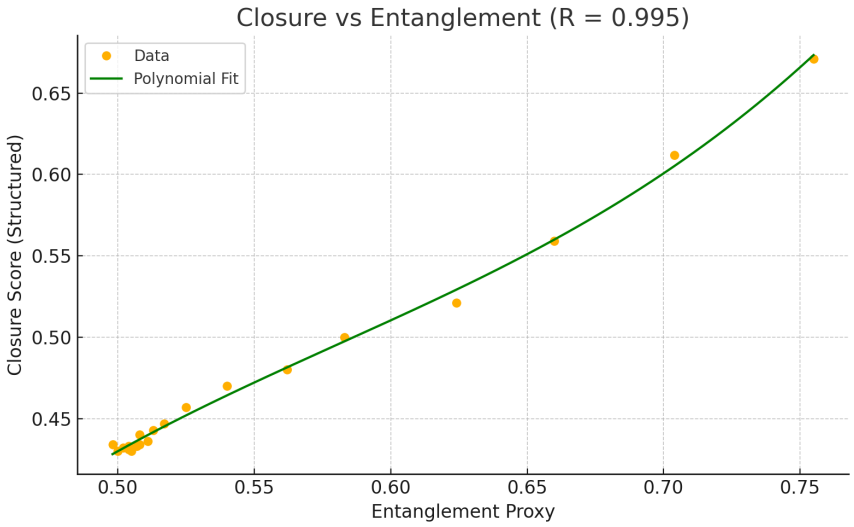

Figure 2: Closure vs. Entanglement for Structured Circuits (R = 0.995).

B Main Python Code Used in Experiments

Listing 1: Core Experiment Script for 2G22-Based Structural Closure Scan

```
# Full code inserted here from previous test (e.g., 2G22 modified version)

import os, math, time
import numpy as np
import matplotlib.pyplot as plt
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit_aer.noise import thermal_relaxation_error, NoiseModel
from qiskit.quantum_info import Statevector
from scipy.stats import bootstrap, pearsonr
from numpy.polynomial import Polynomial

TRIALS_PER_T1 = 50
SHOTS_FOR_ENTANGLEMENT = 8192

# Updated model (adjust plateau to 0.5 empirically)
def enhanced_consciousness_model(tau):
    plateau = 0.50
    return plateau + (0.97 - plateau) * np.exp(-(tau / 600)**2)

def create_biological_noise_model(t1=1e4, t2=1e4, gate_time=10):
```

```
    t2 = min(t2, 2 * t1)
    err_1q = thermal_relaxation_error(t1, t2, gate_time)
    err_2q = thermal_relaxation_error(t1, t2, 3 * gate_time)
    model = NoiseModel()
    model.add_all_qubit_quantum_error(err_1q, ['h', 'x', 's', 't'])
    model.add_all_qubit_quantum_error(err_2q.tensor(err_2q), ['cx'])
    return model

def create_topological_circuit(qps):
    qc = QuantumCircuit(qps)
    for i in range(qps):
        qc.h(i); qc.t(i); qc.cx(i, (i + 1) % qps)
        qc.tdg((i + 1) % qps); qc.cx(i, (i + 1) % qps); qc.s(i)
    for i in range(0, qps - 1, 2):
        qc.crz(np.pi / 8, i, i + 1)
    qc.barrier()
    return qc

def create_classical_control(qps):
    qc = QuantumCircuit(qps)
    for i in range(qps):
        if np.random.rand() > 0.5:
            qc.x(i)
        if i < qps - 1 and np.random.rand() > 0.7:
            qc.cz(i, i + 1)
    qc.barrier()
    return qc

def qrng_bits(n_bits, t1=1e4, use_control=False):
    qps = 5
    qc = create_classical_control(qps) if use_control else create_topological
    qc.measure_all()
    noise_model = create_biological_noise_model(t1)
    sim = AerSimulator(noise_model=noise_model)
    result = sim.run(transpile(qc, sim), shots=math.ceil(n_bits / qps)).resul
    counts = result.get_counts()
    if not counts:
        raise ValueError("No counts returned.")
    bitstrings = list(counts.keys())
    probabilities = np.array(list(counts.values()), dtype=np.float64)
    probabilities /= probabilities.sum()
    samples = np.random.choice(bitstrings, size=math.ceil(n_bits / qps), p=pr
    flat = ''.join(samples)[:n_bits]
    return np.array([int(bit) for bit in flat])
```

```
def analyze_structure(bits):
    changes = np.count_nonzero(np.diff(bits))
    persistence = 1 - (changes / len(bits))
    run_lengths = []
    run = 1
    for i in range(1, len(bits)):
        if bits[i] == bits[i - 1]:
            run += 1
        else:
            run_lengths.append(run); run = 1
    run_lengths.append(run)
    irregularity = np.std(run_lengths) / np.mean(run_lengths) if len(run_leng
    max_stability = max(run_lengths)
    return 0.7 * persistence + 0.2 * min(1.0, max_stability / 50) + 0.1 * (1

def estimate_fast_entanglement(qc, t1, shots=SHOTS_FOR_ENTANGLEMENT):
    noise_model = create_biological_noise_model(t1)
    sim = AerSimulator(noise_model=noise_model)
    qc = qc.copy(); qc.measure_all()
    result = sim.run(transpile(qc, sim), shots=shots).result()
    counts = result.get_counts()
    if not counts:
        raise ValueError("No counts available for entanglement estimation.")
    corr_total, total = 0, 0
    for b, c in counts.items():
        bits = [int(x) for x in b[::-1]]
        matches = sum(bits[i] == bits[i + 1] for i in range(len(bits) - 1))
        corr_total += matches * c
        total += c
    return corr_total / (total * (len(bits) - 1)) if total else 0

def run_ultra_short_scan():
    t1_values = np.logspace(2, 4, 20)
    struct_scores, ctrl_scores, ent_vals = [], [], []
    for t1 in t1_values:
        try:
            print(f"Running T1 = {t1:.1f}ns")
            s_trials = [analyze_structure(qrng_bits(5000, t1)) for _ in range
            c_trials = [analyze_structure(qrng_bits(5000, t1, use_control=Tru
            s_arr = np.array(s_trials)
            c_arr = np.array(c_trials)
            struct_scores.append(np.mean(s_arr))
            ctrl_scores.append(np.mean(c_arr))
            qc_ent = create_topological_circuit(5)
            ent = estimate_fast_entanglement(qc_ent, t1)
```

```
            ent_vals.append(ent)
        except Exception as e:
            print(f"T1={t1:.1f}ns failed: {e}")
            continue
    struct_scores = np.array(struct_scores)
    ctrl_scores = np.array(ctrl_scores)
    ent_vals = np.array(ent_vals)
    if len(ent_vals) > 5:
        try:
            R = pearsonr(struct_scores[:len(ent_vals)], ent_vals)[0]
            print(f"closure-entanglement R = {R:.3f}")
        except:
            print("Not enough data for correlation analysis.")
        for i in range(len(ent_vals)):
            print(f"T1={t1_values[i]:.1f}ns | Struct={struct_scores[i]:.3f} |

if __name__ == '__main__':
    run_ultra_short_scan()
```

Detailed Result Table

T_1 (ns)	Struct	Ctrl	Ent Proxy
100.0	0.671	0.456	0.755
127.4	0.612	0.442	0.704
162.4	0.559	0.434	0.660
206.9	0.521	0.431	0.624
263.7	0.500	0.392	0.583
336.0	0.480	0.455	0.562
428.1	0.470	0.447	0.540
545.6	0.457	0.428	0.525
695.2	0.447	0.417	0.517
885.9	0.443	0.435	0.513
1128.8	0.440	0.491	0.508
1438.4	0.436	0.464	0.511
1833.0	0.434	0.409	0.508
2335.7	0.433	0.406	0.504
2976.4	0.432	0.466	0.502
3792.7	0.433	0.450	0.507
4832.9	0.434	0.457	0.498
6158.5	0.430	0.428	0.505
7847.6	0.431	0.472	0.504
10000.0	0.430	0.411	0.500

Table 1: Closure (Struct), Control (Ctrl), and Entanglement Proxy vs. T1 decoherence time. Strong correlation R = 0.995 observed.